UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/720,506	11/24/2003	Mitica Manu	MSFT-2792/306045	4588

41505 7590 01/31/2008

WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR
2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
01/31/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/720,506	MANU, MITICA	
	Examiner	Art Unit	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *31 October 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,4 and 8-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2,4 and 8-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/ are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the RCE filed on October 31, 2007.

2. **Claims 1, 2, 4, and 8-22** are pending.

3. **Claims 1, 2, 4, 8, 10, 11, 16, 20, and 22** have been amended.

4. **Claims 3 and 5-7** have been cancelled.

5. The objections to Claims 2 and 8-22 are withdrawn in view of Applicant's amendments to the claims.

6. The 35 U.S.C. § 101 rejections of Claims 1-7 are withdrawn in view of Applicant's amendments to the claims.

7. It is noted that Claim 10 contains proposed amendments. However, it still bears the "Previously Presented" status identifier.

Response to Amendment

Claim Objections

8. **Claims 9, 10, and 18** are objected to because of the following informalities:

- **Claim 9** recites the limitation "the block of programming code." Applicant is advised to change this limitation to read "the block of procedural-oriented programming code" for the purpose of providing it with proper explicit antecedent basis.

- **Claim 10** recites the limitation "procedural-oriented source code." Applicant is advised to change this limitation to read "procedural-oriented output source code" for the purpose of keeping the claim language consistent throughout the claims.

- **Claim 18** contains a typographical error: Claim 18 should depend on Claim 17, not Claim 15.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

9. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

10. **Claims 1, 2, 4, and 8-22** are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Claims 1, 8, 9, and 22 recite the limitation of procedural-oriented source/programming code. The subject matter is not properly described in the application as filed, since the specification only discloses the block of code written in C++ (*i.e.*, an object-oriented programming language) *(see Page 9, Paragraph [0038])*. The specification further discloses that the block of code may be written in any programming language *(see Page 9, Paragraph [0038])*, but lacks disclosure on the block of code written in a procedural-oriented programming language. The specification does not indicate that any programming language includes the

procedural-oriented programming language. There is no support in the specification describing the block of code as being written in a procedural-oriented programming language. Because the specification does not adequately support the claimed subject matter, it would not reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Claims 2 and 4 depend on Claim 1 and, therefore, suffer the same deficiency as Claim 1.

Claims 10-21 depend on Claim 8 and, therefore, suffer the same deficiency as Claim 8.

11. **Claims 1, 2, 8-10, and 12-22** are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claims contain subject matter, which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

The claimed invention omits the critical step of "generating procedural-oriented output source code from the functional software model" disclosed to be essential to the invention. The element/step is necessary and must occur in the system/method for the claimed invention to function as intended as described in the specification and/or in the preamble of the claims. A claim which omits matter disclosed to be essential to the invention as described in the specification or in other statements of record may be rejected under 35 U.S.C. 112, first paragraph, as not enabling. *In re Mayhew*, 527 F.2d 1229, 188 USPQ 356 (CCPA 1976). See also MPEP § 2164.08(c). Such essential matter may include missing elements, steps or necessary

structural cooperative relationships of elements described by the Applicant as necessary to practice the invention.

12. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

13. **Claims 1, 2, 8-10, and 12-22** are rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential steps, such omission amounting to a gap between the steps. See MPEP § 2172.01. The omitted step is: generating procedural-oriented output source code from the functional software model. The omitted step is considered to be critical to the claimed invention, since the element/step is necessary and must occur in the system/method for the claimed invention to function as intended as described in the specification and/or in the preamble of the claims.

14. **Claims 1, 4, 10-12, 16, 18, and 20-22** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 1, 4, 10, and 11 recite the limitations "at least one of a plurality of programming languages," "the at least one programming language," "at least one target language," and "the at least one target language," respectively. These claims are rendered indefinite because the generated procedural-oriented output source code must be written in a procedural-oriented

programming language. In the interest of compact prosecution, the Examiner subsequently interprets these limitations as reading "at least one of a plurality of procedural-oriented programming languages," "the at least one procedural-oriented programming language," "at least one procedural-oriented target language," and "the at least one procedural-oriented target language," respectively, for the purpose of further examination.

Claim 4 recites the limitation "each of the at least one programming languages." This is awkward claim language and thus, renders the claim indefinite. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "the at least one procedural-oriented programming language" for the purpose of further examination.

Claims 12, 16, and 20 recite limitations relating to features of the object-oriented programming paradigm (*e.g.*, object, class, etc.). These claims are rendered indefinite because a procedural-oriented programming paradigm does not include these features. In the interest of compact prosecution, the Examiner subsequently does not give any patentable weight to these limitations for the purpose of further examination.

Claim 21 depends on Claim 20 and, therefore, suffers the same deficiency as Claim 20.

Claim 18 recites the limitation "the passive entity." There is insufficient antecedent basis for this limitation in the claim. In the interest of compact prosecution, the Examiner subsequently

interprets this claim as depending on Claim 17 for the purpose of further examination. Thus, such dependency would provide sufficient antecedent basis for this limitation in the claim.

Claim 22 recites the limitation "[a] computer-readable storage medium including computer-readable instructions." The claim is rendered indefinite because computer-readable instructions can only be stored or recorded on a computer-readable medium. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "[a] computer-readable storage medium storing computer-readable instructions" for the purpose of further examination.

Claim Rejections - 35 USC § 103

15. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

16. **Claims 8, 9, 12, 15, 16, 19, and 20** are rejected under 35 U.S.C. 103(a) as being unpatentable over **US 6,502,239 (hereinafter "Zgarba")**.

As per **Claim 8**, Zgarba discloses:

- processing a block of object-oriented programming code and generating from the processed block of object-oriented programming code a functional software model *(see Column*

3: 7-12, "The primary example used to clarify the operation of the embodiment and represented in FIGS. 5-11 Is based around the C++ programming language, although there is no reason the invention could not operate on other object oriented languages or other types of languages, as will become apparent."; Column 5: 1-8, "Reverse engineering according to this embodiment is effected by parsing the code and transforming all the elements of the code which can be represented by the software model into a generic meta-model ..." It is inherent that the source code contains various code blocks.);

- defining a plurality of code elements within the block of object-oriented programming code *(see Figure 3A; Column 3: 30-33, "For example, code exported from a Sybase. PowerBuilder application shown in FIG. 3A includes information concerning the width, height and position of a window type class w_sort.")*;

- specifying a structure of the block of object-oriented programming code including the plurality of code elements *(see Figure 3A; Column 3: 30-33, "For example, code exported from a Sybase. PowerBuilder application shown in FIG. 3A includes information concerning the width, height and position of a window type class w_sort.")*; and

- generating from the plurality of code elements and the structure of the block of object-oriented programming code including the plurality of code elements a graphical representation of the plurality of code elements and flow of the block of object-oriented programming code *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components." and 49-67 to Column 4: 1-3, "For example, if the*

source code language were C++, the software model might provide for the modeling of C++ classes, structs, unions and enumeration classes with similar structures ... ").

However, Zgarba does not disclose:

- procedural-oriented programming code; and
- processing a block of procedural-oriented programming code from an innermost element to an outermost element.

Official Notice is taken that it is old and well-known within the computing art to write the source code in a procedural-oriented programming language. Zgarba discloses that the source code can be written in other types of languages *(see Column 3: 7-12)*. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to write the source code in a procedural-oriented programming language instead of an object-oriented programming language. The modification would be obvious because one of ordinary skill in the art would be motivated to generate a software model from a block of procedural-oriented source code.

Official Notice is also taken that it is old and well-known within the computing art to process source code from an innermost element to an outmost element. When source code is parsed, it is either parsed from the innermost element to the outermost element or from the outermost element to the innermost element. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include processing a block of procedural-oriented programming code from an innermost element to an outermost element. The modification would be obvious because one of ordinary skill in the art would be motivated to process all the code elements of the source code.

As per **Claim 9**, the rejection of **Claim 8** is incorporated; and Zgarba further discloses:

- receiving the definition of the plurality of code elements with the block of procedural-oriented programming code and specifying the structure of the block of procedural-oriented programming code via a user interface *(see Column 3: 30-33, "For example, code exported from a Sybase. PowerBuilder application shown in FIG. 3A includes information concerning the width, height and position of a window type class w_sort.").*

As per **Claim 12**, the rejection of **Claim 8** is incorporated; and Zgarba further discloses:

- wherein one of the plurality of code elements comprises a variable, comment, constant, object, function, method, prototype, member, data type, callback, delegate, reference, field, variant, property, interface, class, type, enumeration, structure, primitive, array, or event handle *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components.").*

As per **Claim 15**, the rejection of **Claim 8** is incorporated; and Zgarba further discloses:

- wherein one of the plurality of code elements comprises an evaluation entity *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in*

the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components.").

As per **Claim 16**, the rejection of **Claim 15** is incorporated; and Zgarba further discloses:

- wherein the evaluation entity comprises one of a method call, a plurality of code entities, a plurality of code relations, or an instantiation of a class *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components.").*

As per **Claim 19**, the rejection of **Claim 8** is incorporated; and Zgarba further discloses:

- wherein one of the plurality of code elements comprises a block entity *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components.").*

As per **Claim 20**, the rejection of **Claim 19** is incorporated; and Zgarba further discloses:

- wherein the block entity comprises a method entity, a member entity, a class entity, a namespace entity, or a file entity *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling*

for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components.").

17. **Claims 1, 2, 4, 10, 11, 17, 18, 21, and 22** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Zgarba** in view of **US 6,199,195 (hereinafter "Goodwin")**.

As per **Claim 1**, Zgarba discloses:

- a modeler for defining at least one of a plurality of code elements and a structure of a code block and generating a graphical representation of the at least one code element and structure of the code block, wherein the modeler processes input comprising a code block of object-oriented source code and generates from the input a code model comprising a graphical representation of a structure and flow of the code block *(see Column 3: 7-12, "The primary example used to clarify the operation of the embodiment and represented in FIGS. 5-11 Is based around the C++ programming language, although there is no reason the invention could not operate on other object oriented languages or other types of languages, as will become apparent." and 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components." and 49-67 to Column 4: 1-3, "For example, if the source code language were C++, the software model might provide for the modeling of C++ classes, structs, unions and enumeration classes with similar structures ..."; Column 5: 1-8, "Reverse engineering according to this embodiment is effected by parsing the code and transforming all the elements*

of the code which can be represented by the software model into a generic meta-model ... " It is inherent that the source code contains various code blocks.).

However, Zgarba does not disclose:

- a computer display;
- procedural-oriented source code;
- wherein the modeler processes input comprising a code block of procedural-oriented source code from an innermost element to an outermost element; and
- a selector for selecting at least one of a plurality of procedural-oriented programming languages in which to generate procedural-oriented output source code from the functional model.

Official Notice is taken that it is old and well-known within the computing art to write the source code in a procedural-oriented programming language. Zgarba discloses that the source code can be written in other types of languages *(see Column 3: 7-12)*. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to write the source code in a procedural-oriented programming language instead of an object-oriented programming language. The modification would be obvious because one of ordinary skill in the art would be motivated to generate a software model from a block of procedural-oriented source code.

Official Notice is also taken that it is old and well-known within the computing art to process source code from an innermost element to an outmost element. When source code is parsed, it is either parsed from the innermost element to the outermost element or from the outermost element to the innermost element. Therefore, it would have been obvious to one of

ordinary skill in the art at the time the invention was made to include wherein the modeler processes input comprising a code block of object-oriented source code from an innermost element to an outermost element. The modification would be obvious because one of ordinary skill in the art would be motivated to process all the code elements of the source code.

Goodwin discloses:

- a computer display *(see Figure 1: 110)*; and
- a selector for selecting at least one of a plurality of object-oriented programming languages in which to generate object-oriented output source code from the functional model *(see Column 13: 45-55, "Thus, when the code generator 330 is instantiated (Block, 500, FIG. 5) a selection (Block 502, FIG. 5) is made as to whether a local (IDL) data model source is being used or a schema server data model is being used." and "Thus, the code generator 330 can support the creation of, for example, IDL, JAVA or C++ files (CORBA, Java RMI, and/or COM) based on certain user preferences and selections.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include a computer display; and a selector for selecting at least one of a plurality of object-oriented programming languages in which to generate object-oriented output source code from the functional model. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a different programming language from the software model.

Official Notice is also taken that it is old and well-known within the computing art to generate procedural-oriented source code from a model. Goodwin discloses that the code

generator can support the creation of various files in different programming languages based on user selections *(see Column 13: 52-55)*. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to generate the file in a procedural-oriented programming language instead of an object-oriented programming language. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a procedural-oriented programming language from the software model.

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and Zgarba further discloses:

- a user interface for receiving the definition of the at least one code element and the structure of the code block *(see Column 3: 30-33, "For example, code exported from a Sybase. PowerBuilder application shown in FIG. 3A includes information concerning the width, height and position of a window type class w_sort.")*.

As per **Claim 4**, the rejection of **Claim 1** is incorporated; however, Zgarba does not disclose:

- a code generator for receiving the graphical representation of the at least one code element and the structure of the code block and the at least one procedural-oriented programming language and generating procedural-oriented output source code in the at least one procedural-oriented programming language.

Goodwin discloses:

- a code generator for receiving the graphical representation of the at least one code element and the structure of the code block and the at least one procedural-oriented programming

language and generating procedural-oriented output source code in the at least one procedural-oriented programming language *(see Column 13: 45-55, "Thus, when the code generator 330 is instantiated (Block, 500, FIG. 5) a selection (Block 502, FIG. 5) is made as to whether a local (IDL) data model source is being used or a schema server data model is being used." and "Thus, the code generator 330 can support the creation of, for example, IDL, JAVA or C++ files (CORBA, Java RMI, and/or COM) based on certain user preferences and selections.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include a code generator for receiving the graphical representation of the at least one code element and the structure of the code block and the at least one procedural-oriented programming language and generating procedural-oriented output source code in the at least one procedural-oriented programming language. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a procedural-oriented programming language from the software model.

As per **Claim 10**, the rejection of **Claim 8** is incorporated; however, Zgarba does not disclose:

- specifying at least one procedural-oriented target language in which procedural-oriented output source code for the graphical representation is to be generated.

Goodwin discloses:

- specifying at least one procedural-oriented target language in which procedural-oriented output source code for the graphical representation is to be generated *(see Column 13:*

45-55, "Thus, when the code generator 330 is instantiated (Block, 500, FIG. 5) a selection (Block 502, FIG. 5) is made as to whether a local (IDL) data model source is being used or a schema server data model is being used." and "Thus, the code generator 330 can support the creation of, for example, IDL, JAVA or C++ files (CORBA, Java RMI, and/or COM) based on certain user preferences and selections.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include specifying at least one procedural-oriented target language in which procedural-oriented output source code for the graphical representation is to be generated. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a procedural-oriented programming language from the software model.

As per **Claim 11**, the rejection of **Claim 10** is incorporated; however, Zgarba does not disclose:

- generating the procedural-oriented output source code in the at least one procedural-oriented target language.

Goodwin discloses:

- generating the procedural-oriented output source code in the at least one procedural-oriented target language *(see Column 13: 45-55, "Thus, when the code generator 330 is instantiated (Block, 500, FIG. 5) a selection (Block 502, FIG. 5) is made as to whether a local (IDL) data model source is being used or a schema server data model is being used." and*

"Thus, the code generator 330 can support the creation of, for example, IDL, JAVA or C++ files (CORBA, Java RMI, and/or COM) based on certain user preferences and selections.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include generating the procedural-oriented output source code in the at least one procedural-oriented target language. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a procedural-oriented programming language from the software model.

As per **Claim 17**, the rejection of **Claim 8** is incorporated; however, Zgarba does not disclose:

- wherein one of the plurality of code elements comprises a passive entity.

Goodwin discloses:

- wherein one of the plurality of code elements comprises a passive entity *(see Column 8: 49-53, "Also shown are a plurality of model adapters 310 for defining a translation of the logical models of the modeling tools 302, 304, 406 into unified models, expressed in a unified modeling language, such as Unified Modeling Language (UML).").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include wherein one of the plurality of code elements comprises a passive entity. The modification would be obvious because one of ordinary skill in the art would be motivated to model all aspects of a software program.

As per **Claim 18**, the rejection of **Claim 17** is incorporated; however, Zgarba does not disclose:

- wherein the passive entity comprises a comment or a modeling diagram.

Goodwin discloses:

- wherein the passive entity comprises a comment or a modeling diagram *(see Column 8: 49-53, "Also shown are a plurality of model adapters 310 for defining a translation of the logical models of the modeling tools 302, 304, 406 into unified models, expressed in a unified modeling language, such as Unified Modeling Language (UML).")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include wherein the passive entity comprises a comment or a modeling diagram. The modification would be obvious because one of ordinary skill in the art would be motivated to model all aspects of a software program.

As per **Claim 21**, the rejection of **Claim 20** is incorporated; however, Zgarba does not disclose:

- wherein a many-to-many relationship exists between block entities.

Goodwin discloses:

- wherein a many-to-many relationship exists between block entities *(see Column 4: 31-36, "A 'relationship' defines a link between two object classes." and "Relationships can be one-to-one, one-to-many, or many-to-many.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include wherein a many-to-many relationship exists between block entities. The modification would be obvious because one of ordinary skill in the art would be motivated to link classes to other classes.

Claim 22 is a computer-readable storage medium claim corresponding to the method claim above (Claim 8) and, therefore, is rejected for the same reason set forth in the rejection of Claim 8.

18. **Claims 13 and 14** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Zgarba** in view of **US 6,684,385 (hereinafter "Bailey")**.

As per **Claim 13**, the rejection of **Claim 8** is incorporated; however, Zgarba does not disclose:

- wherein one of the plurality of code elements comprises a code relation.

Bailey discloses:

- wherein one of the plurality of code elements comprises a code relation *(see Column 8: 30-36, "... each program object typically performs some useful function, such as a Boolean operation (e.g., AND, OR, etc.), a mathematical operation, a data acquisition operation ..., renders some comparison (e.g., less than, greater than, equal to, etc.), and so on.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bailey into the teaching of Zgarba to include wherein one of the plurality of code elements comprises a code relation. The modification would be obvious because one of ordinary skill in the art would be motivated to model all aspects of a software program.

As per **Claim 14**, the rejection of **Claim 13** is incorporated; however, Zgarba does not disclose:

- wherein the code relation comprises a mathematical operator.

Bailey discloses:

- wherein the code relation comprises a mathematical operator *(see Column 8: 30-36, "... each program object typically performs some useful function, such as a Boolean operation (e.g., AND, OR, etc.), a mathematical operation, a data acquisition operation ..., renders some comparison (e.g., less than, greater than, equal to, etc.), and so on.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bailey into the teaching of Zgarba to include wherein the code relation comprises a mathematical operator. The modification would be obvious because one of ordinary skill in the art would be motivated to model all aspects of a software program.

Response to Arguments

19. Applicant's arguments with respect to Claims 1, 8, and 22 have been considered, but are moot in view of the new ground(s) of rejection.

Conclusion

20. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR

system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

QC
January 8, 2008

WEI ZHEN
SUPERVISORY PATENT EXAMINER